SECURED ENCRYPTED COMMUNICATIONS IN A VOICE BROWSER

Inventor(s): Brett Gavagni
Bruce D. Lucas

International Business Machines Corporation

IBM Docket No. BOC9-2000-0014
IBM Disclosure No. BOC8-2000-0019

Express Mail Label No. EK575132635US

BACKGROUND OF THE INVENTION

Technical Field

This invention relates to the field of computer audio user interfaces and more particularly to a system and method for performing secured communications between a voice browser and a server.

Description of the Related Art

Privacy in data communications has become a significant issue with the exponential increase in e-commerce transactions on the Internet. Typically a client computer and server computer require that exchanged information remain private to both parties. For instance, in an online banking transaction, the client requires that the sharing of the client's account number and password include only the intended bank and no other party. Presently, privacy in data transactions can be secured only in selected applications protocols through the use of security technologies which can incorporate either asymmetric, symmetric or a combination of asymmetric and symmetric encryption algorithms. The Secured Sockets Layer ("SSL") protocol represents one such security technology which incorporates both asymmetric and symmetric encryption algorithms.

SSL is a transport-layer protocol that can be established between a client and a server. SSL is typically integrated directly with selected underlying application protocols. For example, the Hypertext Transfer Protocol ("HTTP") has been successfully integrated with SSL. Specifically, HTTP packets are encapsulated in SSL packets and are transported over TCP/IP. HTTP integrated with SSL is commonly referred to as "HTTPS" and can be used to securely view and exchange Web-based content encoded in hypertext markup language ("HTML"). Other protocols integrated with SSL include Telnet, the File Transfer Protocol ("FTP"), the Lightweight Directory Access Protocol ("LDAP"), the Internet Message Access Protocol ("IMAP"), and the Network News Transfer Protocol ("NNTP").

SSL is intended to provide a secure pipe between a client and a server. SSL is

session-oriented and can maintain state, despite the execution of SSL over such protocols as HTTP which, in of itself, is stateless. Finally, SSL provides privacy through encryption, both asymmetric and symmetric, authentication based upon certificates, a vehicle for authorization through SSL's support for certificates, integrity by incorporating hash functions, and digital signing as part of the transport protocol.

Briefly, in an SSL compliant visual Web browser executing the "HTTPS" protocol, an SSL session can be established when a client selects a uniform resource locator ("URL") referencing a server compliant with the HTTPS protocol. The server can respond by delivering to the client, an X.509 certificate containing a distinguished name referencing a Certificate Authority ("CA") and a public key. The client can examine the server certificate by referencing the issuing CA and can verify the integrity of the server certificate if the issuing CA is configured in the visual Web browser as trustworthy. Subsequently, the server can perform optional client authentication by requesting a certificate from the client. The server, too, can examine the client certificate by referencing the issuing CA and can verify the integrity of the client certificate if both the client and the issuing CA are configured in the server as trustworthy. If the server successfully authenticates the client certificate, the SSL session can continue. Otherwise , the session can be terminated.

Subsequently, the client can "challenge" the server using asymmetrical encryption technology in order to verify that the server indeed possesses the private key associated with the public key contained in the server certificate. In challenging the server, the client can generate a random string of data and can encrypt the random string of data using the server's public key contained in the server certificate. The client can transmit the encrypted data to the server and can request that the server deliver the data to the client. In order to deliver the data to the client, however, the server first must decrypt the data using the server's private key which corresponds to the server's public key contained in the server certificate. Optionally, the server, too, can challenge the client using a similar exchange of encrypted data.

Once the client and server have been mutually authenticated, the client and the server can agree upon a shared secret for use in future symmetrical encryption and decryption operations. Typically, the client can select the secret and encrypt the selected secret using the server's public key. The client can transmit the asymmetrically encrypted secret to the server so that only the client and the server share the common secret. When both the client and the server have agreed upon the shared secret, symmetrical data transfer can begin between the client and the server using the shared secret as the key to the symmetrical encryption and corresponding decryption operations. Notably, a more thorough treatment of the SSL protocol has been published by Netscape Communications Corporation of Mountain View, California in Freier, Karlton, Kocher, *The SSL Protocol Version 3.0* (Netscape Communications Corp. March 1996), incorporated herein by reference. Additionally, an SSL 3.0 compatible standard has been approved by the Internet Engineering Task Force ("IETF") and has been published by the IETF as Dierks & Allen, *RFC2246 - The TLS Protocol Version 1.0* (IETF January 1999), incorporated herein by reference.

Unlike visual Web browsers executing the HTTPS protocol, SSL has not been integrated with Voice Browsers. Generally, a Voice Browser, unlike a visual Web browser, does not permit a user to interact with Web-based content visually. Rather, a Voice Browser, which can operate in conjunction with a Speech Recognition Engine and Speech Synthesis Engine, can permit the user to interact with Web-based content audibly. That is, the user can provide voice commands to navigate from Web-based document to document. Likewise, Web-based content can be presented to the user audibly, typically in the form of speech synthesized text. Thus, Voice Browsers can provide voice access and interactive voice response to Web-based content and applications, for instance by telephone, personal digital assistant, or desktop computer.

Significantly, Voice Browsers can be configured to interact with Web-based content encoded in VoiceXML. VoiceXML is a markup language for distributed voice applications based on extended markup language ("XML"), much as HTML is a markup

language for distributed visual applications. VoiceXML is designed for creating audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and Dual Tone Multifrequency ("DTMF") key input, recording of spoken input, telephony, and mixed-initiative conversations. Version 1.0 of the VoiceXML specification has been published by the VoiceXML Forum in the document Linda Boyer, Peter Danielsen, Jim Ferrans, Gerald Karam, David Ladd, Bruce Lucas and Kenneth Rehor, *Voice eXtensible Markup Language (VoiceXML™) version 1.0*, (W3C May 2000), incorporated herein by reference. Additionally, Version 1.0 of the VoiceXML specification has been submitted to the World Wide Web Consortium by the VoiceXML Forum as a proposed industry standard.

Version 1.0 of the VoiceXML specification provides a high-level programming interface to speech and telephony resources for application developers, service providers and equipment manufacturers. As noted in W3C submission, standardization of VoiceXML will simplify creation and delivery of Web-based, personalized interactive voice-response services; enable phone and voice access to integrated call center databases, information and services on Web sites, and company intranets; and help enable new voice-capable devices and appliances. Still, the VoiceXML specification lacks a mechanism for secure communications through encrypted network transmissions via the SSL protocol over the TCP/IP protocol. Accordingly, what is needed is a Voice Browser incorporating SSL support for performing secure communications in a data communications network.

SUMMARY OF THE INVENTION

The present invention is a Voice Browser for processing VoiceXML encoded Web content through a secure connection established using symmetric and asymmetric encryption techniques. In the preferred embodiment, the symmetric and asymmetric encryption techniques are included in a Java implementation of the SSL 3.0 protocol for providing secure communications through encrypted network transmissions between a VoiceXML-compliant Voice Browser Server and a network device. Specifically, the method of the present invention can authenticate the network device and negotiate a shared secret between the client and the server using asymmetrical encryption techniques. Subsequently, the method of the present invention can facilitate secure communications between the client and the server of data in a VoiceXML document using symmetrical encryption techniques.

The method of the invention can include the steps of transmitting a request to the network device to establish a secured communication session between the Voice Browser and the network device and authenticating the network device. Subsequent to the authentication, a shared secret can be negotiated between the network device and the Voice Browser. Once a shared secret has been negotiated, VoiceXML-based Web content can be encrypted using the shared secret as an encryption key. Additionally, the encrypted VoiceXML-based Web content can be exchanged between the network device and the Voice Browser. Finally, the VoiceXML-based Web content can be decrypted using the shared secret as a decryption key. Significantly, the Voice Browser can be a VoiceXML Browser Server.

The step of authenticating the network device can include transmitting a digital certificate from the network device to the Voice Browser and validating the certificate authority. The digital certificate can have a public key and a reference to a certificate authority. Specifically, the digital certificate can be an X.509-compliant digital certificate. Optionally, the method can further include the step of authenticating the Voice Browser. The step of authenticating the Voice Browser can include transmitting a

digital certificate from the Voice Browser to the network device and validating the certificate authority. As before, the digital certificate can have a public key and a reference to a certificate authority. Specifically, the digital certificate can be an X.509-compliant digital certificate.

The step of authenticating the network device can further include the step of challenging the network device. Likewise, the step of authenticating the Voice Browser can further include the step of challenging the Voice Browser. The step of challenging the network device can include encrypting a message using the public key contained in the digital certificate; transmitting the encrypted message from the Voice Browser to the network device; decrypting the encrypted message using a private key corresponding to the public key; and, transmitting the decrypted message to the Voice Browser. Similarly, the step of challenging the Voice Browser can include encrypting a message using the public key contained in the digital certificate; transmitting the encrypted message from the network device to the Voice Browser; decrypting the encrypted message using a private key corresponding to the public key; and, transmitting the decrypted message to the network device.

In the preferred embodiment, the negotiating step can include the steps of: generating a key for use in a symmetric cryptographic algorithm; encrypting the generated key with the public key; transmitting the encrypted key to the network device; and, decrypting the key in the network device with a private key corresponding to the public key. Alternatively, the negotiating step can include the steps of: generating a key for use in a symmetric cryptographic algorithm; encrypting the generated key with the public key; transmitting the encrypted key to the Voice Browser; and, decrypting the key in the Voice Browser with a private key corresponding to the public key.

In the preferred embodiment, the method of the present invention can further include the steps of: exchanging a list of supported symmetrical cryptographic algorithms for the network device and the Voice Browser; selecting a symmetrical cryptographic algorithm from the list; and, performing the encrypting and decrypting

steps using the selected symmetrical cryptographic algorithm.

A method for performing secured communications in a Voice Browser can include the steps of: transmitting a request from the Voice Browser to a network device for a secure communications session between the Voice Browser and the network device; receiving from the network device a digital certificate containing a public key and a reference to a certificate authority; and, authenticating the network device based on the digital certificate. Preferably, the digital certificate can be an X.509-compliant digital certificate.

Subsequent to the authentication, the method can include the steps of negotiating a shared secret with the network device; encrypting data using the shared secret as an encryption key and transmitting the encrypted data to the network device; and, receiving encrypted Web content from the network device and decrypting the Web content using the shared secret as a decryption key. Significantly, the Web content can be a VoiceXML document and the Voice Browser can be a VoiceXML Browser Server.

In the preferred embodiment, the transmitting step can further include the steps of: transmitting to the network device a list of supported encryption algorithms for use in the encryption and decryption steps. Notably, the network device can select an encryption algorithm from among the list. Subsequently, the data can be encrypted using the selected encryption algorithm and the Web content can be decrypted using the encryption algorithm.

BRIEF DESCRIPTION OF THE DRAWINGS

There are presently shown in the drawings embodiments which are presently preferred, it being understood, however, that the invention is not limited to the precise arrangements and instrumentalities shown.

Fig. 1 is an illustration of the establishment of a secured communications session between a Voice Browser and a network device.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is a Voice Browser enabled to perform secured communications with a network device. In particular, the Voice Browser can request a secured connection with a network device. Subsequently, the Voice Browser can receive a response from the network device in which the network device can acknowledge the request of the Voice Browser. Upon receiving the acknowledgment from the network device, the Voice Browser can authenticate the network device in order to ensure the identity of the network device. If the Voice Browser determines the identity of the network device to be authentic, the Voice Browser and the network device can select a shared secret to be used as an encryption key during an ensuing secured communications session. Finally, the Voice Browser and the network device can perform secured communications using the shared secret as an encryption key.

Advantageously, the secured communications functionality provided to the Voice Browser can be the result of the combination of a secured communications interface in the Voice Browser and a platform-independent, standards-based implementation of the Secured Sockets Layer ("SSL") secured communications protocol. In the preferred embodiment, the standards-based, platform-independent implementation of the SSL protocol is the SSLite for Java™ SSL implementation library manufactured by IBM Corporation of Armonk, New York. A class hierarchy for SSLite is attached hereto in Appendix A. Additionally, Javadocs documentation for each class listed in Appendix A are provided in Appendixes B-E. Specifically, Appendix B describes the class HttpsURLConnection, Appendix C describes the class HttpsClient, Appendix D describes the class HttpsURLStreamHandlerFactory and Appendix E describes the class HttpsURLStreamHandler. Still, the invention is not limited in regard to the particular secured communications library to be combined with the secured communications interface. In particular, the present invention can also incorporate the Transport Layer Security ("TLS") protocol defined by the Transport Layer Security Working Group of the Internet Engineering Task Force, the Kerberos protocol

developed by the Massachusetts Institute of Technology, or other suitable secured communications protocols.

Figure 1 illustrates a simplified approach to SSL secured communications between a Voice Browser and a network device. As shown in Figure 1, First, for an SSL connection to become established between the Voice Browser and the network device, an SSL handshake is performed. Specifically, the Voice Browser can transmit to the network device a “client hello” message. The client hello message can include a request for a connection with the network device in addition to the capabilities of the client, for example the preferred secured communications protocol, the cipher suites available to perform encryption and supported data compression methods. The network device can acknowledge the client hello message with a “server hello” message which can include a cipher suite selected from the cipher suites listed in the client hello message, and a compression method selected from the list of supported encryption methods. Notably, if the network device is unable to support any of the encryption algorithms contained in the cipher suite provided by the Voice Browser, the network device can notify the Voice Browser that the handshake attempt has failed. Subsequently, the connection between the Voice Browser and the network device can be closed.

Still, if handshake attempt is successful, the network device can transmit to the Voice Browser a digital certificate which can contain the network device’s public key in addition to a reference to a certificate authority which acts as a trusted repository for digital certificates. The Voice Browser can authenticate the digital certificate by verifying that the certificate authority is a trusted repository of digital certificates. If the Voice Browser can successfully authenticate the certificate authority, a secure connection can be established. Notably, the network device can optionally authenticate the Voice Browser in the same the Voice Browser authenticates the network device. Notwithstanding, mutual authentication is not required in the present invention and the scope of the present invention is not to be limited in this regard.

Once the authentication process has been completed, the Voice Browser can transmit a "ClientKeyExchange" message to the network device. Specifically, the ClientKeyExchange message is a shared secret which has been encrypted using the public key of the network device, received in the digital certificate of the network device. The shared secret can be a randomly generated key for use in a symmetrical encryption algorithm. Despite the generation of the random key, however, the network device still preferably verifies that an identical key is not already in use with another client, be it another Voice Browser or other client application. If the network device determines that the key is already in use, the network device can notify the Voice Browser that another key must be generated. Notably, the invention is not limited in regard to the mechanism for generating a key. Rather, the key can be predetermined and stored in a database, generated according to a pre-defined algorithm, or other suitable key generation or key selection method.

When the Voice Browser and the network device have agreed upon a shared secret, specifically a common symmetric key for encrypting subsequently communications, both the Voice Browser and the network device can exchange a "ChangeCipherSpec" message confirming that both are ready to begin secured communications. Subsequently, the Voice Browser and the network device can begin secure communications using a symmetrical encryption algorithm with the shared secret as the encryption key.

With regard to the particular implementation of the present invention in which the SSL secured communications protocol is combined with secured communications interface of the Voice Browser, the SSL secured communications library can contain an "HttpsURLConnection" object which can provide methods for performing secured communications with HTTP servers. A complete description of the HttpsURLConnection class is included in the Javadoc "Class HttpsURLConnection" attached hereto as Appendix B. As is apparent from the class hierarchy of Appendix A, the HttpsURLConnection class is derived from the Java extension HttpURLConnection.

Accordingly, the class HttpsURLConnection is a platform-independent, standards-based implementation of the SSL protocol.

The following is a source code listing for a preferred interface between the Voice Browser and the secured communications protocol, specifically SSL. As is apparent from the source code, the following steps are minimally performed in order to establish a secured connection to a network device. First, a URL object is defined and instantiated for a fully-qualified URL. Concurrently, a stream handler is established for handling data streams received from the fully-qualified URL. Second, an unsecured connection is established with a network device addressed by the URL in which the symmetrical encryption algorithm can be specified in addition to the compression method. Also, the authentication process can be performed and a shared secret negotiated. Third, a secure connection can be established using the shared secret as a key to the selected encryption method.

```
import java.io.*;
import java.util.*;
import java.net.*;
import com.ibm.speech.net.www.protocol.https.*;

//begin class VoiceXMLBrowserServer
public class VoiceXMLBrowserServer
{
  public static void main(String args[])
  {
    try
    {
      String fullQualURL = new String ("https://www.ibm.com/software/speech/vxmlpage.vxml");

      URL url;

      URL.setURLStreamHandlerFactory(new HttpsURLStreamHandlerFactory());

      //Create file for inputstream dump
      FileOutputStream fout = new FileOutputStream("fetched.vxml");

      //Create URL object
      url = new URL(fullQualURL);

      //Setup Connection
      HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
```

```
//SSL Implementation Specific API Extensions.
//Optional usage, defaults included with implementation distribution
//(Documented usage in Javadoc API definition)
conn.setKeyRingDatabase("ralvs6");      //Set keyring database to use.
                                        //Default is specified as provided with implementation.
conn.setTimeout(3);                     //Set connection timeout in seconds.
conn.setAsyncConnections = true;        //Set SSL messages to be processed asynchronously
                                        //by a dedicated thread.
conn.setEnabledCompressionMethods("IBM_ZIP_SPEED");     //Set compression method.
conn.setEnabledCipherSuites("  SSL_RSA_WITH_RC4_128_MD5
                               SSL_RSA_WITH_RC4_128_SHA");

//Set enabled cipher suites (Encryption Algorithms)

HttpURLConnection API (Standard Java platform networking API)
conn.setRequestMethod("POST");
conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
conn.setRequestProperty("accept", "text/vxml");

//Initiate secure connection
conn.connect();

//Get inputstream and do something with it
if (conn.getInputStream()!=null)
{
   BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
   String line;
   while ((line = in.readLine()) != null)
   {
      line = line + "\n";
      fout.write(line.getBytes());
   }
}
//Close connection
conn.disconnect();
    }
    catch (Exception e)
    {
       System.out.println("Error: " + e.getMessage());
       e.printStackTrace();
    }
  }
}
//end class VoiceXMLBrowserServer
```

The preferred design and implementation of the present invention can be performed entirely in the Java programming language so as to avoid platform

dependencies. As such, the preferred design and implementation of the present invention is an "Optional Package". Optional packages , formerly known as "standard" extensions or "extensions" are packages of Java classes and associated native code that application developers can use to extend the functionality of the core platform. The extension mechanism allows a Java virtual machine (VM) to use the optional-package classes in much the same way as the VM uses bootstrap classes. Like bootstrap classes, classes in optional packages do not have to be placed on the class path. Also, the extension mechanism provides a method for needed optional packages to be retrieved from specified URLs when they are not already installed in the Java 2 Runtime Environment or Java 2 SDK.

Optional packages are embodied in JAR files, in which every JAR file is a potential optional package. A JAR file can be made to play the role of an optional package in two ways: First, by being placed in a special location in the Java 2 Runtime Environment or Java 2 SDK directory structure - in which case it is an "installed" optional package; and second, by being referenced in a specified way from the manifest of the JAR file of an applet or application - in which case it is a "download" optional package. When the VM is searching for a class of a particular name, it will first look among the bootstrap classes. If it fails to find the desired class there, it will next look for the class among any installed optional packages. If it doesn't find the class among either the bootstrap classes or the installed optional packages, the VM will search among any download optional packages referenced by the application or applet. The VM only searches the class path if it fails to find a class among the bootstrap classes or optional package classes.

The preferred embodiment of the present invention provides an abstraction of the underlying complicated key exchange, handshake and encrypted data transmission associated with secure data communications. In consequence, a Voice Browser in accordance with the inventive arrangements can access the abstracted method of the present invention through a reference to a library including an implementation of the

performing secured data communications. Hence, the present invention addresses the problem of secured communications in a Voice Browser by providing a Voice Browser incorporating SSL support for performing secure communications in a data communications network.

The method of the invention can be realized in hardware, software, or a combination of hardware and software. Machine readable storage according to the present invention can be realized in a centralized fashion in one computer system, or in a distributed fashion where different elements are spread across several interconnected computer systems. Any kind of computer system or other apparatus adapted for carrying out the methods described herein is acceptable. A typical combination of hardware and software could be a general purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein. The present invention can also be embedded in a computer program product which comprises all the features enabling the implementation of the methods described herein, and which when loaded in a computer system is able to carry out these methods. A computer program in the present context can mean any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following: (a) conversion to another language, code or notation; and (b) reproduction in a different material form.